Summary

(a) To avoid repetition of code and bulky programs functionally related statements are isolated into a function.

(b) Function declaration specifies what is the return type of the function and the types of parameters it accepts.

(c) Function definition defines the body of the function.

(d) Variables declared in a function are not available to other functions in a program. So, there won't be any clash even if we give same name to the variables declared in different functions.

(e) Pointers are variables which hold addresses of other variables.

(f) A function can be called either by value or by reference.

(g) Pointers can be used to make a function return more than one value simultaneously.

(h) Recursion is difficult to understand, but in some cases offer a better solution than loops.

(i) Adding too many functions and calling them frequently may slow down the program execution.

Exercise

Simple functions, Passing values between functions

[A] What would be the output of the following programs:

(a)
```
main( )
{
    printf ( "\nOnly stupids use C?" ) ;
    display( ) ;
}
display( )
{
    printf ( "\nFools too use C!" ) ;
    main( ) ;
}
```

OUTPUT

Only stupids use C?
Fools too use C!

INFINITE LOOP

```
(b)  main( )
     {
          printf ( "\nC to it that C survives" ) ;
          main( ) ;
     }
```

infinite loop
output
C to it that C survives

```
(c)  main( )
     {
          int  i = 45, c ;
          c = check ( i ) ;
          printf ( "\n%d", c ) ;
     }
     check ( int  ch )
     {
          if ( ch >= 45 )
               return ( 100 ) ;
          else
               return ( 10 * 10 ) ;
     }
```

```
(d)  main( )
     {
          int  i = 45, c ;
          c = multiply ( i * 1000 ) ;
          printf ( "\n%d", c ) ;
     }
     check ( int  ch )
     {
          if ( ch >= 40000 )
               return ( ch / 10 ) ;
          else
               return ( 10 ) ;
     }
```

[B] Point out the errors, if any, in the following programs:

```
(a)  main( )
     {
```

```
     int  i = 3, j = 4, k, l ;
     k = addmult ( i, j ) ;
     l = addmult ( i, j ) ;
     printf ( "\n%d %d", k, l ) ;
}
addmult ( int  ii, int  jj )
{
     int  kk, ll ;
     kk = ii + jj ;
     ll = ii * jj ;
     return ( kk, ll ) ;
}
```

(b)
```
main( )
{
     int  a ;
     a = message( ) ;
}
message( )
{
     printf ( "\nViruses are written in C" ) ;
     return ;
}
```

(c)
```
main( )
{
     float  a = 15.5 ;
     char  ch = 'C' ;
     printit ( a, ch ) ;
}
printit ( a, ch )    not defined the data type
{
     printf ( "\n%f %c", a, ch ) ;
}
```

(d)
```
main( )
{
     message( ) ;
```

```
    message( ) ;
}
message( ) ;
{
    printf ( "\nPraise worthy and C worthy are synonyms" ) ;
}
```

(e)

```
main( )
{
    let_us_c( )
    {
        printf ( "\nC is a Cimple minded language !" ) ;
        printf ( "\nOthers are of course no match !" ) ;
    }
}
```

(f)

```
main( )
{
    message( message ( ) ) ;
}
void message( )
{
    printf ( "\nPraise worthy and C worthy are synonyms" ) ;
}
```

[C] Answer the following:

(a) Is this a correctly written function:

```
sqr ( a ) ;
int  a ;                            no
{
    return ( a * a ) ;
}
```

(b) State whether the following statements are True or False:

1. The variables commonly used in C functions are available to all the functions in a program. ✓

2. To return the control back to the calling function we must use the keyword **return**.

3. The same variable names can be used in different functions without any conflict. ✓

4. Every called function must contain a **return** statement. ✗

5. A function may contain more than one **return** statements. ✓

6. Each **return** statement in a function may return a different value. ✓

7. A function can still be useful even if you don't pass any arguments to it and the function doesn't return any value back. ✓

8. Same names can be used for different functions without any conflict. ✓

9. A function may be called more than once from any other function. ✓

10. It is necessary for a function to return some value. ✗

[D] Answer the following:

(a) Write a function to calculate the factorial value of any integer entered through the keyboard.

(b) Write a function **power (a, b)**, to calculate the value of **a** raised to **b**.

(c) Write a general-purpose function to convert any given year into its roman equivalent. The following table shows the roman equivalents of decimal numbers:

Decimal	**Roman**	**Decimal**	**Roman**
1	i	100	c
5	v	500	d
10	x	1000	m
50	l		

Example:

Roman equivalent of 1988 is mdccccclxxxviii
Roman equivalent of 1525 is mdxxv

(d) Any year is entered through the keyboard. Write a function to determine whether the year is a leap year or not.

(e) A positive integer is entered through the keyboard. Write a function to obtain the prime factors of this number.

For example, prime factors of 24 are 2, 2, 2 and 3, whereas prime factors of 35 are 5 and 7.

Function Prototypes, Call by Value/Reference, Pointers

[E] What would be the output of the following programs:

(a)
```
main( )
{
    float  area ;
    int  radius = 1 ;
    area = circle ( radius ) ;
    printf ( "\n%f", area ) ;
}
circle ( int  r )
```

```
{
    float  a ;
    a = 3.14 * r * r ;
    return ( a ) ;
}
```

(b)
```
main( )
{
    void  slogan( ) ;
    int  c = 5 ;
    c = slogan( ) ;
    printf ( "\n%d", c ) ;
}
void slogan( )
{
    printf ( "\nOnly He men use C!" ) ;
}
```

[F] Answer the following:

(a) Write a function which receives a **float** and an **int** from **main( )**, finds the product of these two and returns the product which is printed through **main( )**.

(b) Write a function that receives 5 integers and returns the sum, average and standard deviation of these numbers. Call this function from **main( )** and print the results in **main( )**.

(c) Write a function that receives marks received by a student in 3 subjects and returns the average and percentage of these marks. Call this function from **main( )** and print the results in **main( )**.

[G] What would be the output of the following programs:

(a)
```
main( )
{
    int  i = 5, j = 2 ;
```

```
        junk ( i, j ) ;
        printf ( "\n%d %d", i, j ) ;
    }
    junk ( int  i, int  j )
    {
        i = i * i ;
        j = j * j ;
    }
```

(b)
```
    main( )
    {
        int  i = 5, j = 2 ;
        junk ( &i, &j ) ;
        printf ( "\n%d %d", i, j ) ;
    }
    junk ( int  *i, int  *j )
    {
        *i = *i * *i ;
        *j = *j * *j ;
    }
```

(c)
```
    main( )
    {
        int  i = 4, j = 2 ;
        junk ( &i, j ) ;
        printf ( "\n%d %d", i, j ) ;
    }
    junk ( int  *i, int  j )
    {
        *i = *i * *i ;
        j = j * j ;
    }
```

(d)
```
    main( )
    {
        float  a = 13.5 ;
        float  *b, *c ;
        b = &a ;  /* suppose address of a is 1006 */
```

```
     c = b ;
     printf ( "\n%u %u %u", &a, b, c ) ;
     printf ( "\n%f %f %f %f %f", a, *(&a), *&a, *b, *c ) ;
}
```

[H] Point out the errors, if any, in the following programs:

(a)
```
main( )
{
     int  i = 135, a = 135, k ;
     k = pass ( i, a ) ;
     printf ( "\n%d", k ) ;
}
pass ( int  j, int  b )
int  c ;
{
     c = j + b ;
     return ( c ) ;
}
```

(b)
```
main( )
{
     int  p = 23, f = 24 ;
     jiaayjo ( &p, &f ) ;
     printf ( "\n%d %d", p, f ) ;
}
jiaayjo ( int  q, int  g )
{
     q = q + q ;
     g = g + g ;
}
```

(c)
```
main( )
{
     int  k = 35, z ;
     z = check ( k ) ;
     printf ( "\n%d", z ) ;
}
```

```
check ( m )
{
    int  m ;
    if ( m > 40 )
         return ( 1 ) ;
    else
         return ( 0 ) ;
}
```

(d)
```
main( )
{
    int  i = 35, *z ;
    z = function ( &i ) ;
    printf ( "\n%d", z ) ;
}
function ( int  *m )
{
    return ( m + 2 ) ;
}
```

[I] What would be the output of the following programs:

(a)
```
main( )
{
    int  i = 0 ;
    i++ ;
    if ( i <= 5 )
    {
         printf ( "\nC adds wings to your thoughts" ) ;
         exit( ) ;
         main( ) ;
    }
}
```

(b)
```
main( )
{
    static int  i = 0 ;
    i++ ;
    if ( i <= 5 )
    {
        printf ( "\n%d", i ) ;
        main( ) ;
    }
    else
        exit( ) ;
}
```

[J] Attempt the following:

(a) A 5-digit positive integer is entered through the keyboard, write a function to calculate sum of digits of the 5-digit number:

(1) Without using recursion
(2) Using recursion

(b) A positive integer is entered through the keyboard, write a program to obtain the prime factors of the number. Modify the function suitably to obtain the prime factors recursively.

(c) Write a recursive function to obtain the first 25 numbers of a Fibonacci sequence. In a Fibonacci sequence the sum of two successive terms gives the third term. Following are the first few terms of the Fibonacci sequence:

1 1 2 3 5 8 13 21 34 55 89...

(d) A positive integer is entered through the keyboard, write a function to find the binary equivalent of this number using recursion.

(e) Write a recursive function to obtain the running sum of first 25 natural numbers.

(f) Write a C function to evaluate the series

$$\sin(x) = x - (x^3 / 3!) + (x^5 / 5!) - (x^7 / 7!) + \cdots$$

to five significant digits.

(g) Given three variables **x**, **y**, **z** write a function to circularly shift their values to right. In other words if x = 5, y = 8, z = 10 after circular shift y = 5, z = 8, x =10 after circular shift y = 5, z = 8 and x = 10. Call the function with variables **a**, **b**, **c** to circularly shift values.

(h) Write a function to find the binary equivalent of a given decimal integer and display it.

(i) If the lengths of the sides of a triangle are denoted by **a**, **b**, and **c**, then area of triangle is given by

$$area = \sqrt{S(S-a)(S-b)(S-c)}$$

where, S = (a + b + c) / 2

(j) Write a function to compute the distance between two points and use it to develop another function that will compute the area of the triangle whose vertices are **A(x1, y1)**, **B(x2, y2)**, and **C(x3, y3)**. Use these functions to develop a function which returns a value 1 if the point **(x, y)** lines inside the triangle ABC, otherwise a value 0.

(k) Write a function to compute the greatest common divisor given by Euclid's algorithm, exemplified for J = 1980, K = 1617 as follows:

1980 / 1617 = 1	1980 – 1 * 1617 = 363
1617 / 363 = 4	1617 – 4 * 363 = 165
363 / 165 = 2	363 – 2 * 165 = 33
5 / 33 = 5	165 – 5 * 33 = 0

Thus, the greatest common divisor is 33.